## **K-NEAREST NEIGHBOR (KNN) CLASSIFICATION FOR DIABETES PREDICTION - TASK 2**

****

|  |  |
| --- | --- |
| **NAME** | ABDUL MUIZZ, MUHAMMAD ABDULLAH |
| **REGISTRATION NUMBER** | FA23-BCS-117, FA23-BCS-108 |
| **DATE OF SUBMISSION** | 26TH SEPTEMBER 2025 |
| **MODERATOR** | SIR UMAR NAUMAN |

**Table of Contents:**

[**K-NEAREST NEIGHBOR (KNN) CLASSIFICATION FOR DIABETES PREDICTION - TASK 2** 1](#_Toc209709187)

[**Abstract:** 3](#_Toc209709188)

[**Problem Statement:** 4](#_Toc209709189)

[**Dataset Description:** 5](#_Toc209709190)

[**Key Features Used:** 5](#_Toc209709191)

[**Data Preprocessing:** 5](#_Toc209709192)

[**Methodology:** 6](#_Toc209709193)

[**1. K-Nearest Neighbor (KNN) Algorithm** 6](#_Toc209709194)

[**2. K-Value and Distance Metric** 6](#_Toc209709195)

[**3. Evaluation Metrics** 6](#_Toc209709196)

[**Results and Analysis:** 7](#_Toc209709197)

[**Classification Results:** 7](#_Toc209709198)

[**Analysis of Findings:** 7](#_Toc209709199)

[**List of Figures:** 9](#_Toc209709200)

[**Conclusion:** 10](#_Toc209709201)

[**Limitations:** 10](#_Toc209709202)

[**Future Work:** 10](#_Toc209709203)

[**References:** 11](#_Toc209709204)

## **Abstract:**

This project implements the K-Nearest Neighbor (KNN) classification algorithm to predict the onset of diabetes based on diagnostic medical attributes. The analysis was performed on the Pima Indians Diabetes Dataset, which includes several physiological measurements. The primary objective was to build a reliable classifier by performing essential data preprocessing, including feature scaling, and then training and evaluating the KNN model. The model was configured with k=11 neighbors and used the Euclidean distance metric. Upon testing, the classifier achieved a predictive accuracy of 74.7%, demonstrating its effectiveness as a baseline model for this medical classification task.

## **Problem Statement:**

Diabetes is a prevalent chronic disease, and its early detection is crucial for effective management and prevention of complications. Medical datasets often contain valuable patterns that can be uncovered using machine learning.

This project aims to:

* Apply the K-Nearest Neighbor (KNN) algorithm to classify patients as either diabetic or non-diabetic.
* Preprocess the dataset by scaling features to ensure the model's performance is not biased by variable magnitudes.
* Train the KNN classifier on a portion of the dataset.
* Evaluate the model's performance on unseen test data using key metrics like the confusion matrix and accuracy score.

## **Dataset Description:**

**Dataset:** diabetes.csv (Pima Indians Diabetes Dataset)  
**Total Records:** 768  
**Original Features:** 9 columns

### **Key Features Used:**

The model was trained on 8 independent features to predict the single target variable.

* Pregnancies: Number of times pregnant
* Glucose: Plasma glucose concentration
* BloodPressure: Diastolic blood pressure (mm Hg)
* SkinThickness: Triceps skin fold thickness (mm)
* Insulin: 2-Hour serum insulin (mu U/ml)
* BMI: Body mass index
* DiabetesPedigreeFunction: A function that scores the likelihood of diabetes based on family history
* Age: Age in years
* Outcome (Target Variable): Class variable (0 for non-diabetic, 1 for diabetic)

### **Data Preprocessing:**

Before model training, the following preprocessing steps were performed:

1. **Feature-Target Split:** The dataset was separated into features (X) and the target variable (y).
2. **Train-Test Split:** The data was divided into an 80% training set and a 20% testing set to ensure the model could be evaluated on unseen data.
3. **Feature Scaling:** **StandardScaler** was applied to normalize the features. This is a critical step for distance-based algorithms like KNN, as it prevents features with larger scales (e.g., Glucose) from disproportionately influencing the distance calculations compared to features with smaller scales (e.g., DiabetesPedigreeFunction).

## **Methodology:**

### **1. K-Nearest Neighbor (KNN) Algorithm**

KNN is a non-parametric, supervised learning algorithm used for both classification and regression. For classification, a data point is assigned to the class that is most common among its **k-nearest neighbors**. The "closeness" of neighbors is determined by a distance metric.

### **2. K-Value and Distance Metric**

* **Choice of 'k'**: The number of neighbors was set to **k=11**. This value was chosen as an odd number to avoid ties in class voting and is a common starting point to balance model complexity and generalization.
* **Distance Metric**: The **Euclidean distance** (p=2) was used to calculate the distance between data points in the feature space. It is the most common distance metric used with the KNN algorithm.

### **3. Evaluation Metrics**

The model's performance was assessed using two standard metrics:

* **Accuracy Score**: This metric measures the proportion of total predictions that were correct. It is calculated as:

Accuracy = True Positives + True Negatives / Total Data

* **Confusion Matrix**: This table provides a detailed breakdown of the classification results, showing the number of correct and incorrect predictions for each class. It contains four values:
  + **True Positives (TP)**: Correctly predicted positive cases.
  + **True Negatives (TN)**: Correctly predicted negative cases.
  + **False Positives (FP)**: Incorrectly predicted positive cases (Type I error).
  + **False Negatives (FN)**: Incorrectly predicted negative cases (Type II error).

## **Results and Analysis:**

The model was evaluated on the 20% test set, which consisted of 154 samples.

### **Classification Results:**

|  |  |
| --- | --- |
| **Metric** | **Result** |
| Accuracy Score | 74.7% |
| Confusion Matrix | [[88, 14], [25, 27]] |

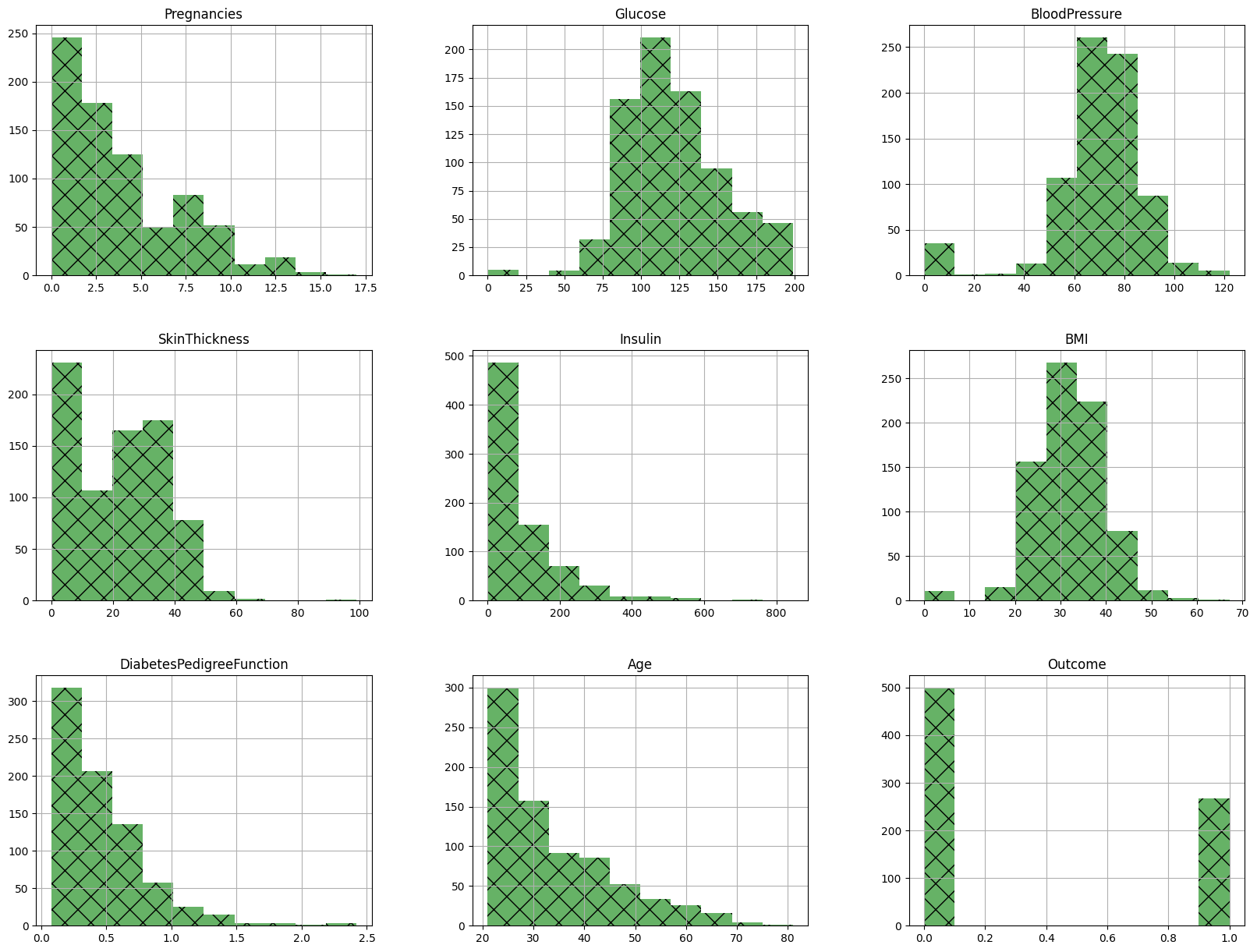


Figure 1: Distribution of Features in the Dataset

### **Analysis of Findings:**

* **Overall Performance**: The model achieved an accuracy of **74.7%**, which means it correctly predicted the outcome for approximately 3 out of every 4 patients in the test set.
* **Confusion Matrix Breakdown**:
  + **True Negatives (TN) = 88**: 88 non-diabetic patients were correctly identified.
  + **False Positives (FP) = 14**: 14 non-diabetic patients were incorrectly identified as diabetic.
  + **True Positives (TP) = 27**: 27 diabetic patients were correctly identified.
  + **False Negatives (FN) = 25**: 25 diabetic patients were incorrectly identified as non-diabetic. This is the most serious type of error in a medical context, as it could lead to a lack of necessary treatment.
* **Key Insight**: While the overall accuracy is reasonable, the model struggles with identifying diabetic patients (high number of False Negatives). This may be due to the simplicity of the model or the class imbalance in the dataset.

## **List of Figures:**

**Figure 1:** Distribution of Features in the Dataset

## **Conclusion:**

The K-Nearest Neighbor algorithm provided a solid baseline for predicting diabetes, achieving **74.7% accuracy** after proper feature scaling. The results show that KNN can effectively identify patterns in medical data but also highlight its limitations, particularly in correctly classifying the minority class (diabetic patients). The model serves as a strong starting point for further analysis and improvement.

### **Limitations:**

* **Choice of 'k'**: The value k=11 was selected manually. A more robust approach, such as cross-validation, could be used to find the optimal 'k' value.
* **Computational Cost**: KNN can be slow and memory-intensive on very large datasets since it needs to store all training data and calculate distances for each new prediction.
* **Curse of Dimensionality**: The model's performance can degrade as the number of features increases.

### **Future Work:**

* **Optimize 'k'**: Implement an automated method (e.g., GridSearchCV or a for-loop with cross-validation) to find the best value for 'k'.
* **Address Class Imbalance**: Use techniques like SMOTE (Synthetic Minority Over-sampling Technique) to create a more balanced training dataset, which could improve the prediction of diabetic cases.
* **Compare Algorithms**: Evaluate other classification models like Logistic Regression, Support Vector Machines (SVM), or Random Forest to see if they can achieve higher accuracy or reduce the False Negative rate.

## **References:**

1. Scikit-learn Developers. (2023). *scikit-learn: Machine Learning in Python*. [scikit-learn.org](https://scikit-learn.org).
2. Smith, J. W., Everhart, J. E., Dickson, W. C., Knowler, W. C., & Johannes, R. S. (1988). *Pima Indians Diabetes Database*. UCI Machine Learning Repository. [https://archive.ics.uci.edu/ml/datasets/pima+indians+diabetes](https://www.google.com/search?q=https://archive.ics.uci.edu/ml/datasets/pima%2Bindians%2Bdiabetes)